INFORMATION TECHNOLOGY, COMPUTER SCIENCE, AND MANAGEMENT

UDC 681.3.06 https://doi.org/10.23947/2687-1653-2020-20-4-422-429

Polynomially computable Σ–specifications of hierarchical models of reacting systems

V. N. Glushkova, K. S. Korovina
Don State Technical University (Rostov-on-Don, Russian Federation)

Introduction. Verification packages design and analyze the correctness of parallel and distributed systems within the framework of various classes of temporal logics of linear and branching time. The paper discusses a polynomially realizable class of $\Delta_0 T$ -formulas interpreted on multi-sorted models with hierarchical suspensions. The suspension structure is described by an arbitrary context-free (CF) grammar. The predicates and functions of the model signature are interpreted on the original CF-list, which is completed during the interpretation process.
Materials and Methods. A constant model is constructed for theories from $\Delta_0 T$-quasiidentities with Noetherian and confluence properties. We consider formulas of the multi-sorted first-order predicate calculus (PC) language with variables of the "list" sort interpreted on models with a hierarchized suspension. The theory is interpreted in terms of grammar inference trees describing the behavior of the specified system. The CF-grammar rules hierarchize the action space of the modeled system. It is noted that the expressive capabilities of $\Delta_0 T$-formulas are insufficient for modeling real-time systems. Therefore, expressions with unbounded universal quantifier ∀, known as PT formulas, are used for the specification.
Results. The logical specification of an automated complex which consists of a workpiece manipulator is given as an example. The location of the positions is fixed by sensors. The operating cycle of the manipulator is described. The specification of its operation consists in the hierarchization of actions by the rules of the CF-grammar and their description by the first-order PT-formulas taking into account the time values.
Discussion and Conclusions. The paper shows that the class of the considered formulas can be used to model real-time systems. An example of the logical specification of a manipulator behavior control device is given.

Keywords*:* logical specification, theory model, reactive system, CF-grammar, first-order PC-formula.

For citation: V. N. Glushkova, K. S. Korovina. Polynomially computable Σ–specifications of hierarchical models of reacting systems. Advanced Engineering Research, 2020, vol. 20, no. 4, p. 422–429. https://doi.org/10.23947/2687-1653-2020-20-4-422-429

© Glushkova V. N., Korovina K. S., 2020

Introduction. Mathematically sound, practically significant methods of verification of complex software and technical systems are based on the apparatus of mathematical logic [1–4]. The technique of applying this approach to various types of real-time reactive systems (communication protocols, control systems, integrated onboard systems of space technology, etc.) is known.

This technique provides verification of model checking systems [5–7]. Numerous verification packages support the design and analysis of the correctness of parallel and distributed systems within various classes of linear and branching time temporal logics: LTL, CTL, TCTL, etc. [8].

To simulate time in these systems, the standard model of the time automaton is used. This is a finite state machine equipped with a special type of variable — local clock. Quantitative analysis of the time characteristics of the system is complicated by complex exponential algorithms for constructing time zones as equivalence classes [9]. Therefore, it is required to develop a more expressive, practically significant specification language to simplify the analysis.

It is proposed to use the language of Σ- specifications for simulation, highlighted in the concept of semantic programming, which is based on the model-theoretic approach [10]. In this case, the 1st order predicate calculus language, extended by axioms for list structure operations can be used to build a formal model of the analyzed system[1,2,3,4].

Materials and Methods. The paper uses terminology of the papers [11, 12]. Let $\mathcal{M}$ be a many-sorted signature model $\sigma = < I, C, F, R>$. Here, I is a set of sorts, including the "list" sort (*list*). C, F, R are sets of constants, functions, and predicates, respectively. All signature symbols have a type. If $f \in F$ is an n-local function, $n \geq 0$, then its type is $<i_1, \dots i_n, i>$, where $i_1, \dots i_n, i \in I$, and $i_1, \dots i_n$ are types of arguments, i is the type of the function value. Similarly, n- local predicate $r \in R$ is of type $< i_1, \dots i_n >$. The model carrier $\mathcal{M}$ is an indexed family of sets $U_j = C_j, j \in I$, where C_j is a set of constants of sort j; f: $U_{i_1} \times \ldots \times U_{i_n} \to U_i$, $r \subseteq U_{i_1} \times \ldots \times U_{i_n}$.

For the model $\mathcal{M}$, a list suspension $D_G(C)$ from the hierarchized CF-lists, whose structure is set by the CF-grammar, is formed over a set of constants C. Here, N, T are sets of nonterminal and terminal symbols. The set $D_G(C)$ is defined as the smallest set of all lists $<t_1, \dots, t_n>$, formed for each rule $A \to X_1 \dots X_n \in P$, $n \geq 1$ as follows: t_i is an arbitrary constant of C_{X_i}, if $X_i \in T$; otherwise, for $X_i \in N$, the element t_i is an arbitrary list of X_i.

Δ_0-formulas are defined in the traditional way as signature formulas σ using all logical connectives ($\neg$, $\wedge$, $\vee$, $\to$) and bounded quantifiers $\forall x \in t$, $\exists x \in t$, $\forall x \subseteq t$, $\exists x \subseteq t$. Here, x is a variable of an arbitrary sort; t is a term of the *list* sort that does not contain x; $\in$ is the list membership relation; $\subseteq$ the nesting relation for lists, defined as $< \alpha_1, \dots, \alpha_n > \subseteq < \alpha_1, \dots, \alpha_n >$, $m \geq n$.

Below, we will use only bounded quantifiers of the form $\forall x \in y$, $\forall x \,\dot{\in}\, y$, where переменная y is a variable of *list* sort, the relation $\dot{\in}$ is transitive closure of the relationship $\in$. Denote the indexed sequence of variables x_i by $\bar{x}$, and the membership relation or its transitive closure — by $\overset{\circ}{\in}$.

Rules of the CF-grammar hierarchize the action space of the simulated system. For reasons of computational efficiency, a class of $\Delta_0 T$-formulas with a "tree" prefix is distinguished. We introduce the relation $\prec$ — to the "left" for the list elements, namely, for the list $< \cdots \alpha, \beta \dots . >$, we consider $\alpha \prec \beta$.

Definition. Δ_0-formula of the form

$$(\forall v_1 \overset{\circ}{\in} r_1) \dots (\forall v_m \overset{\circ}{\in} r_m)(n_1 \prec l_1) \dots (n_p \prec l_p)\ \Phi(\bar{v}, \bar{r}),\ m \geq 1, p \geq 0$$

is called $\Delta_0 T$-formula if $n_j, l_j \in (\bar{v}, \bar{r})$, $1 \leq j \leq p$; for all prefix variables, the following condition is true: $r_{i+1} = r_i$, $1 \leq i < m$ or $r_{i+1} = v_k$, $k \leq i$. If $r_{i+1} = v_k$, then $v_{i+k} \neq v_k$ and $v_{i+k} \neq r_k$ for all $k \leq i$.

It is easy to show that the prefix of $\Delta_0 T$-formula, due to restrictions on variables, can be presented as a tree with root r_1, vertices v_i, r_i and arcs going from vertex r_i to vertex v_i, $1 \leq i \leq m$.

The expressive capabilities of $\Delta_0 T$-formulas are not sufficient for modeling real-time systems that function cyclically for an indefinite period of time. We will use for the specification of the PT-formula with a universal quantifier $\forall$.

Definition. The formula obtained from $\Delta_0 T$-formula through Φ unbounded universal quantification $\forall v\ \Phi(v)$, is called the PT-formula.

The model $\mathcal{M}$ is defined by a theory of quasi-identities of the form:

$$(\forall v_1 \,\dot{\in}\, r_1) \dots (\forall v_m \,\dot{\in}\, r_m)\ (n_1 \prec l_1) \dots (n_p \prec l_p)(\varphi(\bar{v}, \bar{r}) \to \psi(\bar{v}, \bar{r})).$$

---

[1] Glushkova VN. Verification of real-time robotic hierarchical systems. In: Proc. X All-Russian School-Seminar on Mathematical Modeling and Biomechanics in Modern University. Rostov-on-Don; Taganrog: SFU; 2015. P. 32. (In Russ.)

[2] Glushkova VN. Σ-specification of real-time robotic systems. In: Proc. Int. Conf. on Algebra and Logic, Theory and Applications. Krasnoyarsk: SFU; 2016. P. 22–23. (In Russ.)

[3] Glushkova VN. Logical means of diagnosing errors in a hierarchical S-models. In: Proc. Int. Conf. on Algebra and Mathematical Logic. Kazan: KFU; 2011. P. 58–59. (In Russ.)

[4] Glushkova VN. Logical modeling of robotic technological systems. In: Proc. VIII All-Russian School-Seminar. Rostov-on-Don: SFU; 2013. P. 44. (In Russ.)

Here, the formula φ (ψ) is a conjunction of atomic formulas (or their negations) of the form r, $\tau_1=\tau_2$, (f=τ), $f \in F, r \in R$, τ_1, τ are terms of signature σ.

The Int model construction algorithm implements the modus ponens output rule (if φ and φ→ψ, then ψ). The input data for the interpreter is a set of initial values of functions and predicates of the form:
$S_0 = \{p(\bar{c}), f(\bar{c}) = c_{n+1} | p \in P, f \in F\}$, where $\bar{c}$ is a set of constants of n elements, $n \geq 1$.

Axioms (ax) are processed in a certain order, first, with positive occurrences of predicates in the left and right parts of ax, then, with negative occurrence until fixed points of calculations are obtained. The scope of functions and predicates included in the right-hand side of ax, expands when the left-hand side is true. This is because the interpreter sets new values for functions and predicates so that the right-hand side of ax is also true. Let the state S_n of the analyzed system at the n-th step of the calculation contain the values of all predicates and functions of the model signature, and function τ $P(S) \rightarrow P(S)$ in (terminology [3] — predicate converter) reflects the state change when Int interpreter moves from the n-th step of the calculation to n+1. Int interpreter constructs the smallest fixed point μZ for the monotone converter τ on $P(S)$. $\tau(Z) = \cup_i \tau^i(S_0)$, где $\tau^0(Z) = Z$, $\tau^{i+1}(Z) = \tau(\tau^i(Z))$

Formally, functions $f \in F$ and predicate $sp \in R$ are interpreted on the CF-list $tl(n)$, which represents the derivation tree $tr(n)$ in grammar G, where n is the step of Int work. Due to the difficulty of presenting a compact form of the interpretation algorithm on the elements of the CF-list, we first give a verbal explanation of the algorithm, focusing on $tr(n)$ tree. The input data for the model construction algorithm (Int) are as follows: the derivation source tree $tr(0)$ in grammar G, which is expanded under the construction of model $\mathcal{M}$ and $Fact$=S_0. The CF-grammar is used in the process of building the model as follows. First, the rules P hierarchize the space of actions and states of the analyzed system. We assume that the action names represented in the model signature by predicates and the names of the corresponding nonterminal grammar symbols are the same. Secondly, the symbols from alphabet V of the grammar uniquely define the sorts of all elements of the model universe, including lists, which are assigned to the sort defined by the root mark of the corresponding tree. Sorts will be designated mnemonically with initial lowercase characters for the names of nonterminal and terminal grammar symbols with the addition of *s* (*sort*) symbol at the end. The main advantage of CF-grammars is the possibility of using effective syntactically oriented (SO) methods for analyzing the correctness (verification) of the model developed in the theory of syntactic analysis of programming languages.

The interpreter starts by viewing tree $tr(0)$ from the root top to bottom, from left to right. The prefix of all axioms satisfies the constraints $\Delta_0 T$-formulas. Prefix sorts are defined by the symbols of the CF-grammar G. The interpreter selects as constants the truth domains of the predicates included in axiom $ax \in Th$, the constants associated with the vertices of the tree bush, viewing it from top to bottom, from left to right. Moreover, the bush root is marked with the name of the corresponding predicate. To reflect the dependence of the simulated technical system on the sequence of input signals, it is required to complete the source tree $tr(0)$. To this end, the sequence of rules $pr^+(ax) \in P^+$ is attributed to the tree output obtained in the previous step of the algorithm. Moreover, constants from the truth domain of predicate r are used as terminal symbols subordinate to the tree vertex marked with nonterminal symbol r.

We describe the interpretation algorithm Int more formally, without detailing the procedure $Con(Q, Th)$ — obtaining all logical consequences from the set of formulas Q based on the axioms of the theory Th. $Th_0 = S_0$. Theory $Th_{pos} \subseteq Th$ includes only positive occurrences of predicates. $Th_{neg} \subseteq Th$ includes the negative occurrence of predicates on the right side of the axioms. We denote tr_{pos}, tr_{neg} — the derivation trees generated during the interpretation process.

```
$Q$:=$\emptyset$;
$Q'$:=$Th_0$;
while $Q \neq Q'$ do
$Q_{pos} := Q$;
$Q'_{pos} := Q'$;
while $Q_{pos} \neq Q'_{pos}$ do
```

http://vestnik-donstu.ru

```
$Q_{pos} := Q'_{pos}$;
        $Q'_{pos} := Con(Q'_{pos}, Th_{pos})$
end while
return $(Q_{pos}, tr_{pos})$ ;
$Q_{neg} := \emptyset$;
$Q'_{neg} := Q_{pos}$;
while $Q_{neg} \neq Q'_{neg}$ do
$Q_{neg} := Q'_{neg}$;
        $Q'_{neg} := Con(Q'_{neg}, Th_{neg})$
end while
return $(Q_{neg}, tr_{neg})$;
            $Q := Q_{pos}$;
            $Q' := Q_{neg}$
end while
return $(Q_{neg}, tr_{neg})$
```

The verification of model $\mathcal{M}$ consists in checking the properties that the analyzed system should satisfy. We express these properties by arbitrary $\Delta_0 T$-formulas. Using SO-methods of checking formulas, a proof can be constructed in the same way as in [13].

Theorem. Arbitrary $\Delta_0 T$- formula with *m*-bounded generality quantifiers is tested on the CF-list of power *n* in time $O(n^{m+1})$.

The list power tl is equal to the cardinality of the set$\{s \mid s \dot{\in} tl\}$.

The estimate is upper, and it can be lowered to linear if you check the formulas using specific SO-methods of language processing

Research Results. We present a logical specification of an automated complex consisting of a manipulator maintaining a processing line (*tl*) with two positions: loading and unloading of parts (*ld* and *uld* , respectively) [14]. Sensors record the location of positions. The manipulator functions cyclically starting from the loading position.

We present a logical specification of an automated complex consisting of a manipulator maintaining a process line (tl) (ld and uld, respectively) [14].

CYCLE

1. In the initial position *ld* to load the part, the manipulator lifts the electric drive in 4 seconds. It compresses the automated claws and takes the workpiece (2 sec), lowers the electric drive (4 sec) and moves to the right to the machine until the position sensor is triggered *tl*.

2. To install the workpiece on the machine at position *tl*, the manipulator raises the electric drive, unclenches the automated claws (2 sec), lowers the electric drive. Next, the manipulator waits for 4 min, after which it repeats the same procedures as in position *ld*. Then, the manipulator moves to the left to the unloading position until the limit switch is triggered *uld*.

3. In 8 seconds, the part is unloaded on the conveyor. The manipulator moves to the left until the sensor is locked to the loading position *ld*. Further, the process of complex operation is cyclically repeated.

The system specification consists of several levels. The manipulator behavior is determined by the signals of sensors that record its position: *ld*, *uld*, *tl* (¬*ld*, ¬*uld*, ¬*tl*, negation indicates the absence of the corresponding sign). This sequence of signals is represented by the tuple *mc*=<*x*, *y*, *z*...>, where *x* = *ld* (¬*ld*), *y* = *uld* (¬ *uld*), *z*=*tl* (¬ *tl*). It is generated by a finite-state machine with an initial state *x*. This automaton is constructed according to a right-linear grammar with the rules:

$St \rightarrow ld\, St_1 \mid \neg\, ld St_1 \mid \varepsilon$

$St_1 \rightarrow tl\, St_2 \mid \neg\, tl\, St_2$

$St_2 \rightarrow uld\, St \mid \neg\, uld\, St.$

Denote a set of lists, made up of symbol strings generated by this grammar, by *Dsig*.

The external discrete time (variable *n* in the logical specification) is determined by the number of transitions in the automaton. To describe the second level of the manipulator operation, the CF-grammar is used, which indicates the sequence of actions (*Oper*) of the manipulator:

— *L*, *La* — loading the workpiece by the manipulator in position *ld* and *tl,* respectively;
— *Unl*, *Unla* — unloading of the part in position *uld* and *tl*;
— *Mover*, *Movel* — movement of the manipulator to the right and to the left;
— *Lstop*, *Astop*, *Ulstop* — stop of the manipulator in the corresponding position;
— *Exp* — waiting;
— *Cr* — failure of the manipulator control device.

The states of the manipulator (symbol *Pos*) are affected by its actions. In this example, the state is characterized by continuous time *Timec* and discrete *Timed*, given by a natural number. The value of sort *Timec* is segments of the form $< t_1,\ t_2 >$, $t_1,\ t_2$ — constants, and < is replaced by (or [depending on whether the left border is included in the time segment or not, similarly for >.

When specifying the manipulator behavior, you disregard the value of the time of manipulator movement from one position to another (determined by signals from position sensors — input to the manipulator control device). The signals that are sent to the manipulator actuators for the movement and operation of the automated claws are output signals.

Below are the grammar rules *G*.

1. $Start \rightarrow \{Oper\}^*$.
2. $Oper \rightarrow L \mid La \mid Unl \mid Unla \mid Mover \mid Movel \mid Lstop \mid Astop \mid Ulstop \mid Exp \mid Cr$.
3. $L \rightarrow St$.
4. $La \rightarrow St$.
5. $Unl \rightarrow State$.
6. $Unla \rightarrow State$.
7. $Mover \rightarrow State$.
8. $Movel \rightarrow State$.
9. $Lstop \rightarrow State$.
10. $Astop \rightarrow State$.
11. $Ulstop \rightarrow State$.
12. $Exp \rightarrow State$.
13. $Cr \rightarrow State$.
14. $State \rightarrow Timec\ Timed \mid Timed\ Timed$.
15. $Timec \rightarrow Timed \mid (Timed, Timed) \mid [Timed, Timed) \mid (Timed, Timed] \mid [Timed, Timed]$.

Timed — a class of tokens whose values are natural numbers calculated under the interpretation of theory *Th.*

In theory *Th,* variables in formulas are designated mnemonically according to their sort: ρ (*state*) = ρ (*State*), ρ (*oper*) = ρ (*Oper*), ρ (*n*) = ρ (*t*) = ρ (*Timed*), ρ (*ct*) = ρ (*Timec*). Predicates *Ld*, *Tl*, *Unld* are defined on the set *Timed*. *Ld* (*n*) is true if the manipulator is in the loading position. Similarly, for *Tl* (*n*) — at the position of the processing machine, *Uld* (*n*) — at the position of unloading. Let us list the areas of definition of the remaining predicates: *Lstop*, *Astop*, *Ulstop*, *Mover*, *Movel*, $Cr \subseteq Timed \times Timed$; *L*, *La*, *Unl*, *Unla*, $Exp \subseteq Timec \times Timed$. The formulas use the standard functions $head\ (< x_1,...x_n >) = x_1$, $tail(< x_1,...x_n >) = < x_2,...x_n >$ and function *Mc*: $Timed \rightarrow Dc$. Here, *Dc* is a set of lists of sensor signals.

At the initial time *t* = 0, *n* = 1 and at the 1st step of the calculation, predicate *Lstop*(0,1) is executed; *Mc* (1) = *mc*, where $mc \in Dc$. In axioms 1–11, variables *t*, *n* and *ct* are bound by restricted quantifier $\forall\ state \overset{*}{\in} oper$, $\forall\ t,n,\ ct$

http://vestnik-donstu.ru

∈ *state*. In axioms 12–17, variable *n* is bound by restricted quantifier ∀; s_0 = <<<<0,1>>>> — the initial value of the list on which the theory is interpreted. Its list constituents, in order of nesting depth, have the following sorts: ρ (*R*), ρ (*Oper*), ρ (*Lstop*), ρ (0) = ρ (1) = ρ (*Timed*). Tree T_0 corresponds to list s_0 (Fig. 1).

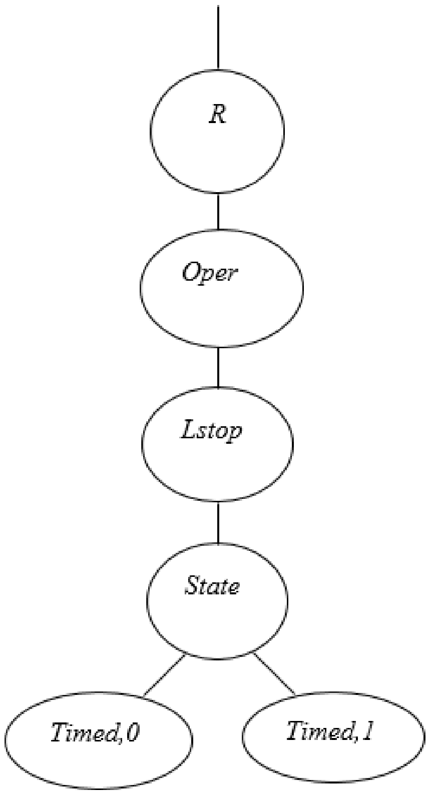

Fig. 1. Tree T_0 corresponding to list s_0

In axioms of the theory, the sequence of grammar rules *G*, that complete tree T_0 , is given in square brackets on the right.

<u>Axioms of the theory</u>

1. *Lstop*(*t*, *n*) ∧ *Ld* (*n*) → *L*([*t*, *t* + 7), *n*) ∧ *Mc* (*n* + 1) = *tail*(*Mc* (*n*)) [1; 2.1; 3; 14.1; 15.3].
2. *L*(*ct*, *n*) ∧ *Tl*(*n* + 1) → *Mover* (*ct* [2], *n* + 1) ∧ *Astop* (*ct* [2], *n* + 1) [1; 2.5; 7; 14.2;1; 2.8; 10; 14.2].
3. *Astop* (*t*, *n*)→ *Unla* ([*t*, *t* + 7], *n*) [1; 2.4; 6; 14.1; 15.2].
4. *Unla* (*ct*, *n*) → *Exp* ((*ct* [2], *ct* [2] + 180), *n*) [1; 2.10; 11; 14.1; 15.2]
5. *Exp* (*ct*, *n*) → *La* ([*ct* [2], *ct* [2] + 3), *n*) ∧ *Mc* (*n* + 1) = *tail*(*Mc* (*n*)) [1; 2.2; 4; 14.1; 15.2].
6. *La* (*ct*, *n*) ∧ *Uld* (*n* + 1) → *Movel* (*ct* [2], *n* + 1) ∧ *Ulstop* (*ct* [2], *n* + 1) [1; 2.6; 8; 14.2; 1; 2.9; 11; 14.2].
7. *Ulstop* (*t*, *n*) → *Unl* ([*t*, *t* + 7), *n*) ∧ *Mc* (*n* + 1) = *tail*(*Mc* (*n*)) [1; 2.3; 5; 14.1; 15.2].
8. *Unl* (*ct*, *n*) ∧ *Ld* (*n* + 1) → *Movel* (*ct* [2], *n* + 1) ∧ *Lstop* (*ct* [2], *n* + 1) [1; 2.6; 8; 14.2; 1; 2.7; 9; 14.2].
9. *Unl* (*ct*, *n*) ∧ ¬ *Ld* (*n* + 1) → *Cr* (*ct* [2], *n* + 1) [1; 2.11; 13; 14.2].
10. *La* (*ct*, *n*) ∧ ¬ *Unld* (*n* + 1)→ *Cr* (*ct* [2], *n* + 1) [1; 2.11; 13; 14.2].
11. *L* (*ct*, *n*) ∧ ¬ *Tl* (*n*+1) → *Cr* (*ct* [2], *n* + 1) [1; 2.11; 13; 14.2].
12. *head* (*Mc* (*n*)) = *ld* → *Ld* (*n*).
13. *head* (*Mc* (*n*)) = ¬ *ld* → ¬ *Ld* (*n*).
14. *head* (*Mc* (*n*)) = *tl* → *Tl* (*n*).
15. *head* (*Mc* (*n*)) = ¬*tl* → ¬ *Tl* (*n*).
16. *head* (*Mc* (*n*)) = *uld* → *Unld* (*n*)
17. *head* (*Mc* (*n*)) = ¬*uld* → ¬ *Uld* (*n*).

Theory *Th* has the Noetherian property, since the change of the variable under the quantifier ∀ is limited to *k* — the number of elements in the source list *mc.* In this case, *head* (*Mc* (*k*+1)) i9s undefined, because *Mc* (*k*+1)=< >. Note that the strings ¬*ld* and others, in the right part of axioms 12–17 have the sort *string*, and ¬ is considered not as a logical operation, but as a symbol.

For the initial value of function *Mc* (1) = <*ld*, *tl*, *uld*, *ld*, ¬ *tl*, *uld* > , we obtain a set of consequences: *Lstop*(0,1), *Ld* (1), *L* ([0,7), 1), *Mc* (2) = < *tl*, *uld*, *ld*, ¬ *tl*, *uld*>, *Mover* (7,2), *Astop* (7,2), *Unla* ([7, 14], 2), *Exp*([14, 194], 2), *La*([194, 197), 2), *Mc* (3) = <*uld*, *ld*, ¬ *tl*, *uld*>, *Movel* (197, 3), *Ulstop* (197, 3), *Unl* ([197, 204], 3), *Mc* (4) = <*ld*, ¬ *tl*, *uld* >, *Movel* (204, 4), *Lstop* (204, 4), *L* ([204, 211), 4), *Mc* (5) = < ¬ *tl*, *uld*>, *Cr* (211, 5).

The resulting set of consequences is hierarchized according to the inference in the grammar *G* obtained as a result of the rules assigned to the interpreted axioms. According to them, 12 more vertices, marked with the same symbol and connected by edges to the root, are added to tree T_0 to the right of the node marked with symbol *Oper*. Subtrees with roots marked with symbols *Ld*, *Mover*, *Astop, etc.,* with their states and constants of sort ρ (*Timed*) obtained as a result of interpretation, are added to the new vertices.

Discussions and Conclusions. On the constructed model, you can check the truth of arbitrary $\Delta_0 T$-formulas. For example, we formalize the statement: “If the manipulator was in the loading position at the instant of time *t* at step *n* of its operation cycle, then at step *n* + 2 after197 sec, it starts unloading for 7 sec”. The formula below is tested on a given list *Oper* of sort ρ (*Oper*):

$$(\forall state \overset{*}{\in} oper)\ (\forall t \in State)\ (\forall n \in state)\ (Lstop(t, n) \rightarrow Unl\ ([t + 197, t + 204], n + 2).$$

References

1. Goguen JA, Meseguer J. Models and equality for logical programming. Lecture Notes in Computer Science. 1987;250:1–22.

2. Kowalski R. Logic for Problem Solving, Revisited. London: Imperial College; 2014. P. 321.

3. Clarke EM, Grumberg O, Peled D. Verifikatsiya modelei programm [Model checking]. Moscow: Moscow Center for Continuous mathematical Education Publ.; 2002. 416 p. (In Russ.)

4. Reps T, Thakur A. Automating Abstract Interpretation. In: 17th International Conference, VMCAI 2016, on Verification, Model Checking and Abstract Interpretation. St. Petersburg, FL, USA, January 17–19, 2016. Paris: Springer; 2016. P. 3–40.

5. Bloem R, Konighofer R, Seidl M. SAT-Based Synthesis Methods for Safety Specs. In: 15th International Conference, VMCAI 2014, on Verification, Model Checking and Abstract Interpretation. San Diego, CA, USA, January 2014. San Diego: Springer; 2014. P. 1–20.

6. Beyer D, Wendler Ph. Reuse of Verification Results. In: 20th International Symposium, SPIN 2013, on Model Checking Software. Stony Brook, July 8–9, 2013. Stony Brook: Springer; 2013. P. 1–15.

7. Alur R, Courcoubetis C, Dill DL. Model-checking for real-time system. Information and Computation. 1993;104(1):2–34.

8. Morbe G, Scholl Ch. Fully Symbolic TCTL Model Checking for Incomplete Timed Systems. In: Proceedings of the Automated Verification of Critical Systems (AVoCS 2013). 2013;66:1–9.

9. D´Silva V. Kroening D, Sousa M. Independence Abstractions and Models of Concurrency. In: 18th International Conference, VMCAI 2017, on Verification, Model Checking and Abstract Interpretation. Paris, France, January 15–17, 2017. Paris: Springer; 2017. P. 149–168.

10. Goncharov SS, Sviridenko DI. Theoretical aspects of Σ-programming. In: Proc. of the International Spring School, April 1985, on Mathematical Methods of Specification and Synthesis of Software Systems' 85. Berlin; Heidelberg: Springer-Verlag; 1985. P. 169–179.

11. Goncharov SS. Modeli dannykh i yazyki ikh opisanii [Data models and their description languages]. Vychislitel'nye sistemy. Logiko-matematicheskie osnovy problemy MOZ. 1985;107:52–70. (In Russ.)

12. Maltsev AI. Algebraicheskie sistemy [Algebraic systems]. Moscow: Nauka; 1976. P. 392. (In Russ.)

13. Glushkova VN. Otsenka slozhnosti realizatsii logicheskikh spetsifikatsii na osnove kontekstno-svobodnykh grammatik [Evaluating the complexity of implementing logical specifications based on context-free grammars]. Cybernetics and Systems Analysis. 1996;4:50–58. (In Russ.)

14. Gorbatov VA, Smirnov MI, Khlytchiev IS. Logicheskoe upravlenie raspredelennymi sistemami [Logical control of distributed systems]. Moscow: Ehnergoatomizdat; 1991. 288 p. (In Russ.)

Submitted 27.07.2020
Scheduled in the issue 05.10.2020

About the Authors:

Glushkova, Valentina N., associate professor of the Mathematics Department, Don State Technical University, (1, Gagarin sq., Rostov-on-Don, 344003, RF), Cand.Sci. (Phys.-Math.), senior researcher, ORCID: https://orcid.org/0000-0003-2719-8590, lar@aaanet.ru

Korovina, Ksenia S., senior lecturer of the Mathematics Department, Don State Technical University, (1, Gagarin sq., Rostov-on-Don, 344003, RF), ORCID: https://orcid.org/0000-0002-1196-3596, ksenichka_@inbox.ru

Claimed contributorship

V. N. Glushkova: formulation of the syntactically oriented hierarchical modeling concept; objectives and tasks of logical specification; academic advising; correction of the conclusions. K. S. Korovina: application of Σ-formulas for the technical system specification; analysis of system verification results and drawing conclusions; the text revision.

All authors have read and approved the final manuscript.